Dependabilidade:

Confiabilidade e Disponibilidade

Prof. Dr. Iaçanã Ianiski Weber

Confiabilidade e Segurança de Software

98G08-4

Gancho para a próxima aula: Safety, Security e Resiliência (atributos e ameaças ampliadas).

Índice

Índice

Por que engenharia de dependabilidade?

- Sistemas reais falham por **causas acidentais** e **causas sistemáticas** (projeto/implementação).
- Em ambientes modernos: **dependências** (serviços, bibliotecas, redes) dominam risco.
- Engenharia precisa de:
  - **Métricas** (quantificar), **modelos** (predizer/planejar),
  - **métodos** (prevenir, tolerar, remover, prever falhas).

Dependabilidade (Dependability): definição e atributos

Definição (visão clássica)

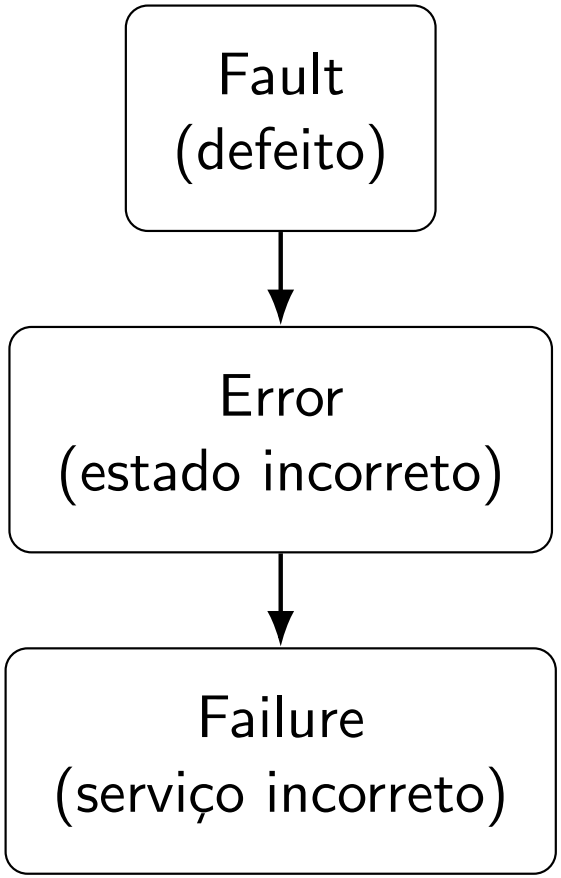
Dependabilidade é a **capacidade de entregar serviço correto e confiável** em que se pode **justificadamente confiar**.

- Atributos (exemplos): **confiabilidade**, **disponibilidade**, **segurança funcional (safety)**, **integridade**, **manutenibilidade**.
- **Security** adiciona preocupações como **confidencialidade** além de disponibilidade/integridade.

Fonte clássica: Avizienis et al., IEEE TDSC (2004).

A cadeia: Fault → Error → Failure

- **Fault (defeito)**: causa interna/externa latente.
- **Error (erro de estado)**: estado interno incorreto gerado por fault ativado.
- **Failure (falha de serviço)**: erro chega à interface de serviço (desvio do especificado).



Reliability vs Availability: intuição correta

Confiabilidade (Reliability)

Probabilidade de **não ocorrer falha** durante um intervalo:
$\mathrm{R}(t) = P(T > t)$.

Disponibilidade (Availability)

Probabilidade de o sistema estar **operacional em um instante** ou em regime estacionário.

- Um sistema pode ser **pouco confiável** (falha com frequência) mas **muito disponível** (recupera rápido).
- E o inverso: falha raramente, mas quando falha, fica indisponível por muito tempo.

Índice

TTF como variável aleatória

Seja T o **tempo até falha** (Time-To-Failure).

$$\mathrm{F}(t) = P(T \leq t), \quad \mathrm{R}(t) = P(T > t) = 1 - \mathrm{F}(t)$$

Densidade:

$$\mathrm{f}(t) = \frac{d\mathrm{F}(t)}{dt}$$

Hazard rate (taxa de risco/instantânea):

$$\mathrm{h}(t) = \frac{\mathrm{f}(t)}{\mathrm{R}(t)}$$

MTTF como área sob $\mathrm{R}(t)$

Para distribuições contínuas (não negativas):

$$\mathrm{MTTF} = \mathbb{E}[T] = \int_0^\infty \mathrm{R}(t)\, dt$$

- Esse resultado é muito útil: você não precisa da densidade explicitamente para obter MTTF.
- Para modelos paramétricos, MTTF vira função direta de parâmetros (ex.: $1/\lambda$).

Modelo exponencial: quando faz sentido?

Hipótese-chave

$\mathrm{h}(t) = \lambda$ constante $\Rightarrow \mathrm{R}(t) = e^{-\lambda t}$ (propriedade sem memória).

- Bom modelo **na região "útil"** da curva **bathtub** (taxa aproximadamente constante).
- Muito usado em testes/planejamento por simplicidade (e por permitir ICs analíticos).

  Referência prática: NIST Reliability/Availability Handbook (exponencial/HPP).

Exponencial: métricas e estimação rápida

$$\mathrm{R}(t) = e^{-\lambda t}, \quad \mathrm{MTTF} = \frac{1}{\lambda}$$

Se em um teste observamos n falhas e TTT = **tempo total em teste** (soma das exposições):

$$\hat{\lambda} = \frac{n}{\mathrm{TTT}}, \quad \widehat{\mathrm{MTTF}} = \frac{\mathrm{TTT}}{n}$$

- TTT funciona bem inclusive com **censura à direita** (itens que não falharam até o fim do teste).

Intervalo de confiança para MTBF/MTTF (exponencial)

No modelo exponencial (HPP), o NIST fornece fatores/tabelas e forma fechada para IC (baseado em χ^2).

- Essencial para não vender "um número mágico" sem incerteza.
- Caso "zero falhas" (muito comum em testes curtos) tem tratamento específico.

Fonte: NIST APR 8.4.5.1 (MTBF e ICs).

Weibull: modelo mais realista para componentes físicos

Weibull (2 parâmetros)

$$\mathrm{R}(t) = \exp\left(-\left(\frac{t}{\eta}\right)^{\beta}\right), \quad \mathrm{h}(t) = \frac{\beta}{\eta}\left(\frac{t}{\eta}\right)^{\beta-1}$$

- $\beta < 1$: "infant mortality" (hazard decrescente).
- $\beta = 1$: exponencial (hazard constante).
- $\beta > 1$: desgaste/envelhecimento (hazard crescente).

Referência didática: ETH Zürich (Reliability Distributions).

Bathtub curve e interpretação de engenharia

- Três fases típicas:
  1. **Mortalidade infantil**: falhas de fabricação/integração (hazard decrescente).
  2. **Vida útil**: falhas aleatórias (hazard constante).
  3. **Desgaste**: envelhecimento (hazard crescente).
- Em **software**, a analogia muda:
  - Sem mudanças: hazard pode ser aproximadamente constante.
  - Com correções/novas features: hazard varia e pode até piorar (regressões).

Índice

MTTF, MTBF, MTTR, MDT (e por que MTTR é "decomponível")

- MTTF: tempo médio até falha (não reparável).
- MTBF: tempo médio entre falhas (reparável).
- MTTR: tempo médio de reparo/recuperação.
- MDT: downtime médio (pode incluir logística, espera, janela, aprovação).

Decomposição útil (Ops/SRE)

$$\mathrm{MTTR} \approx \underbrace{\mathrm{MTTD}}_{\text{detectar}} + \underbrace{\mathrm{MTTI}}_{\text{diagnosticar/decidir}} + \underbrace{\mathrm{MTTM}}_{\text{mitigar/reparar}}$$

Disponibilidade: instantânea vs estacionária

- **Disponibilidade instantânea**: $\mathrm{A}(t) = P(\text{sistema operacional em } t)$.
- **Disponibilidade estacionária**: $\mathrm{A}_\infty = \lim_{t\to\infty} \mathrm{A}(t)$.

Para taxas exponenciais: falha λ e reparo μ,

$$\mathrm{A}_\infty = \frac{\mu}{\lambda + \mu} \quad \Longleftrightarrow \quad \mathrm{A}_\infty = \frac{\mathrm{MTBF}}{\mathrm{MTBF} + \mathrm{MTTR}}$$

“N-nines” e orçamento de indisponibilidade

Meta (%)	Downtime/mês (30d)	Downtime/ano (365d)
99.0	$\approx$ 7h12m	$\approx$ 3d15h
99.9	$\approx$ 43m12s	$\approx$ 8h45m
99.99	$\approx$ 4m19s	$\approx$ 52m
99.999	$\approx$ 26s	$\approx$ 5m15s

- A conta é direta: $\text{downtime} = (1 - \text{meta}) \times \text{janela}$.

Referência prática (error budget / janela mensal): Google SRE (Calculus of Service Availability).

Disponibilidade percebida vs interna (o que você mede?)

- **Disponibilidade interna**: processo/host "up".
- **Disponibilidade do usuário**: requisições boas / total (SLI).
- **Erro parcial**: degradação (latência alta, funcionalidade limitada) pode ser "falha" do ponto de vista do SLO.

Boa prática

Definir SLIs alinhados ao usuário e derivar SLO e "error budget" (1-SLO).

Referência: Google SRE Workbook (alerting e SLOs).

Índice

Assunção de independência: útil, mas perigosa

- Muitas fórmulas clássicas assumem falhas **independentes**.
- Na prática, falhas correlacionadas são comuns:
  - bug em código compartilhado,
  - configuração errada propagada,
  - dependência externa (DNS, IdP, rede),
  - energia/temperatura/ambiente (hardware),
  - ataques (Security).

RBD: sistema em série (dependências "AND")

Se um serviço depende de n componentes independentes:

$$\mathrm{R}_{sys}(t) = \prod_{i=1}^{n} \mathrm{R}_i(t)$$

- Intuição: "uma falha derruba o serviço".
- Em microsserviços, dependências críticas podem dominar a confiabilidade global.

RBD: paralelo ativo (redundância "OR")

Para n componentes em paralelo ativo (1-out-of-n):

$$\mathrm{R}_{sys}(t) = 1 - \prod_{i=1}^{n} \left(1 - \mathrm{R}_i(t)\right)$$

Caso idêntico exponencial ($\mathrm{R}(t) = e^{-\lambda t}$):

$$\mathrm{R}_{sys}(t) = 1 - \left(1 - e^{-\lambda t}\right)^n$$

- Paralelo aumenta confiabilidade (e disponibilidade), **mas** depende de detecção/failover e da ausência de causa comum.

k-out-of-n e votação (ex.: TMR)

Para n componentes e o sistema operar se pelo menos k estiverem OK:

$$\mathrm{R}_{k/n}(t) = \sum_{j=k}^{n} \binom{n}{j} \mathrm{R}(t)^j \left(1 - \mathrm{R}(t)\right)^{n-j}$$

- TMR (Triple Modular Redundancy): $k = 2, n = 3$ com **votador**.
- Atenção: o **votador** vira componente crítico (pode dominar série).

Disponibilidade com redundância: exemplo numérico

Dois servidores idênticos, paralelo ativo, cada um com:

$$\mathrm{MTBF} = 500h, \quad \mathrm{MTTR} = 1h \Rightarrow \mathrm{A} = \frac{500}{501} = 0.9980$$

Assumindo independência e failover "perfeito" (aprox.):

$$\mathrm{A}_{par} \approx 1 - (1 - \mathrm{A})^2$$

- Compare o ganho com o caso em série (dependência "AND") onde a disponibilidade cai.

Common-cause failures (CCF): quando redundância não salva

- Um modelo simples: β-**factor**
  - fração β das falhas é comum a todos os canais (derruba todos ao mesmo tempo);
  - fração $1 - \beta$ é independente.
- Implicações:
  - redundância sem diversidade pode ter ganho muito menor que o esperado.
  - práticas: diversidade de versão, zona/região, fornecedor, config, caminho de rede.

Referência didática: ETH Zürich (Dependent Failures).

Índice

Quando RBD não basta

RBD é ótimo para estrutura "estática", mas falha ao modelar:

- **reparo** (volta do componente ao sistema),
- **standby** (frio/morno/quente),
- **cobertura de detecção** (coverage),
- **tempos não desprezíveis** de failover, reboot, rollback.

Ferramenta padrão

Cadeias de Markov (CTMC) para disponibilidade/confiabilidade em sistemas reparáveis.

Exemplo CTMC: 1-out-of-2 com reparo

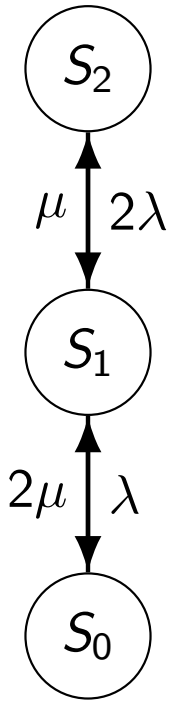
Estados:

- S_2: 2 operacionais
- S_1: 1 operacional
- S_0: 0 operacionais (down)

Transições (aprox.):

- falhas: $S_2 \rightarrow S_1$ taxa 2λ, $S_1 \rightarrow S_0$ taxa λ
- reparos: $S_1 \rightarrow S_2$ taxa μ, $S_0 \rightarrow S_1$ taxa 2μ

Referência de curso: ETH Zürich (Markov Models).



Como extrair disponibilidade estacionária do CTMC

- Resolva $\pi Q = 0$ com $\sum \pi_i = 1$ (distribuição estacionária).
- Disponibilidade: $\mathrm{A}_\infty = 1 - \pi_{S_0}$.

Mensagem prática

Modelos de estado permitem incorporar:

- **tempo de failover**, **coverage**, **reparo limitado** (fila), **standby**.

Índice

Meios para atingir dependabilidade (classificação clássica)

1. **Fault prevention**: evitar introduzir faults (processo, métodos formais, boas práticas).
2. **Fault removal**: encontrar e remover faults (testes, revisão, análise estática).
3. **Fault tolerance**: manter serviço correto apesar de faults (redundância, recuperação).
4. **Fault forecasting**: estimar incidência/consequências (modelagem, testes, dados de campo).

Fonte: Avizienis et al. (2004) e materiais de cursos europeus de dependability.

Observabilidade e MTTR: o “ciclo” operacional

- **Detectar** rápido (reduz MTTD): métricas, logs estruturados, traces, alertas.
- **Diagnosticar** (reduz MTTI): correlação, runbooks, grafos de dependência.
- **Mitigar** (reduz MTTM): rollback automático, feature flags, auto-healing.

Impacto direto

Mesmo com MTBF igual, reduzir MTTR melhora A de forma significativa.

SLO/Error Budget: "engenharia orientada a metas"

- Defina SLI: proporção de eventos bons / total.
- Defina SLO (meta interna) e SLA (contratual).
- **Error budget** $= 1 - \mathrm{SLO}$: "quanto você pode errar" na janela.

Regra importante

Você é tão disponível quanto a soma do risco das suas dependências.

Fonte: Google SRE (The Calculus of Service Availability; SRE Book/Workbook).

Tolerância a falhas: padrões de software (com armadilhas)

- **Timeouts** e **retries** (com jitter/backoff) para falhas transientes.
- **Circuit breaker** para evitar falha em cascata.
- **Bulkheads** (isolamento) para conter propagação.
- **Graceful degradation**: modos de serviço parcial.

Índice

Software Reliability Growth: por que difere de hardware?

- Em hardware, desgaste físico justifica hazard crescente (Weibull $\beta > 1$).
- Em software, falhas tendem a vir de **faults latentes** ativados por certos perfis de uso.
- Em testes/campo, correções removem faults $\Rightarrow$ taxa de falha pode **decrescer** ao longo do tempo.

Exemplo clássico: NHPP (Goel–Okumoto)

Modelo assume que o número cumulativo de falhas segue um processo de Poisson não homogêneo:

$$\mu(t) = N\left(1 - e^{-bt}\right)$$

onde N é o número esperado total de falhas "eventuais" e b controla a rapidez do decaimento.

- Bom para dados de teste onde falhas vão diminuindo com correções.
- Precisa cuidado: pressupõe condições relativamente estáveis e coleta consistente.

Fonte: notas de aula universitárias (Software Reliability Estimation, Univ. of Denver).

Musa–Okumoto (logarítmico) e leitura crítica

Intuição: consertos tardios produzem menor ganho; intensidade decresce com falhas acumuladas.

- Modelos SRGMs são úteis para **planejamento** e **comparação**, não como "verdade absoluta".
- Sempre verifique:
  - viés de coleta (mudança de carga/perfil),
  - mudanças arquiteturais,
  - qualidade do processo de correção (debugging imperfeito),
  - dependências externas (que mudam).

Fonte: Univ. of Denver (W63) e literatura clássica de SRGM.

Índice

Exemplo 1: serviço embarcado de telemetria (com MTTR decomposto)

Dados:

$$\mathrm{MTBF} = 500h, \quad \mathrm{MTTD} = 10min, \quad \mathrm{MTTI} = 20min, \quad \mathrm{MTTM} = 30min$$

$$\mathrm{MTTR} \approx 60min = 1h \Rightarrow \mathrm{A} = \frac{500}{501} = 99.80\%$$

- Se automatizar detecção e rollback: $\mathrm{MTTD} = 1min$, $\mathrm{MTTM} = 5min$ $\Rightarrow \mathrm{MTTR} \downarrow$ forte.

Exemplo 2: watchdog + estado seguro + perda de serviço

- Fault de firmware leva a deadlock.
- Watchdog reinicia em 5s, mas:
  - reinit de periféricos (mais 10s),
  - ressincronização de estado (mais 15s).
- Downtime efetivo por evento: 30s.

Pergunta de engenharia

O que é "falha" para o usuário? Reiniciar em 5s pode ser ótimo, mas perder estado pode ser inaceitável em alguns domínios.

Exemplo 3: SLO 99.9% e dependências (orçamento por componente)

- SLO do serviço A: 99.9% $\Rightarrow$ budget = 0.1% por janela.
- Se A tem N dependências críticas, o budget precisa ser **particionado** (heurística comum).
- Resultado prático: dependências precisam ter "mais 9s" de disponibilidade que o serviço final.

Fonte: Google SRE (The Calculus of Service Availability).

Índice

Exercícios 1: confiabilidade e hazard

1. Para $\lambda = 2 \times 10^{-4}\, h^{-1}$, calcule $\mathrm{R}(100h)$ e MTTF.
2. Uma Weibull com $\beta = 2$, $\eta = 1000h$: calcule $\mathrm{R}(500h)$ e interprete o hazard.
3. Mostre (algebricamente) que hazard constante implica $\mathrm{R}(t) = e^{-\lambda t}$.

Exercícios 2: disponibilidade e trade-offs

1. Um sistema tem $\mathrm{MTBF} = 1200h$ e $\mathrm{MTTR} = 4h$. Qual A?
2. Se você puder escolher entre:
   - reduzir λ em 10% (melhorando robustez), ou
   - reduzir MTTR em 50% (melhorando observabilidade/rollback),

   qual ação dá maior ganho em A neste caso?
3. Calcule o downtime mensal permitido para 99.95% e compare com 99.9%.

Exercícios 3: composição e dependências

1. Dois componentes em série com $\mathrm{A}_1 = 0.999$ e $\mathrm{A}_2 = 0.9995$. Aproxime A_{sys}.
2. Dois componentes em paralelo ativo (independentes) com $\mathrm{A} = 0.999$. Aproxime A_{par}.
3. Com causa comum $\beta = 0.05$, discuta qualitativamente por que o ganho do paralelo cai.

Índice

Resumo

- **Confiabilidade** modela "não falhar no intervalo"; **disponibilidade** inclui **reparo** e **recuperação**.
- Exponencial é um **caso** (hazard constante); Weibull e hazard ajudam a capturar realidade.
- Redundância melhora muito, mas **dependências** e **common-cause** podem dominar.
- Em sistemas modernos, **MTTR** é alavanca enorme via observabilidade e automação.
- Metas SLO/error budget conectam modelagem com operação contínua e trade-offs de mudança.

Gancho: próxima aula (Safety, Security e Resiliência)

- A taxonomia de dependabilidade inclui **Safety** e **Security** como atributos/ameaças complementares.
- **Safety**: falhas podem causar dano físico $\Rightarrow$ foco em hazard, risco, estados seguros, certificação.
- **Security**: faults podem ser **maliciosos** (ataques) $\Rightarrow$ integridade/confidencialidade + disponibilidade sob ataque.
- **Resiliência**: capacidade de **manter e recuperar** serviço sob falhas/ataques/degradação e dependências.

Referências essenciais (para leitura e credibilidade)

- Avizienis, Laprie, Randell, Landwehr — *Basic Concepts and Taxonomy of Dependable and Secure Computing* (IEEE TDSC, 2004).
  `https://www.landwehr.org/2004-aviz-laprie-randell.pdf`
- ETH Zürich — *Reliability of Technical Systems* (distributions, dependent failures, Markov).
  `https://www.ises.ethz.ch/education/lectures/reliability-of-technical-systems.html`
- NIST — *Reliability/Availability Handbook* (exponencial/HPP, MTBF e intervalos de confiança).
  `https://www.itl.nist.gov/div898/handbook/apr/`
- MIT (notes/OCW) — materiais de confiabilidade/bathtub e fundamentos.
  `https://web.mit.edu/2.611/www/notes/BB.pdf`
- University of Pisa — slides de Dependability (cadeia fault-error-failure, métodos).
  `https://docenti.ing.unipi.it/~a008669/`
- Google SRE — *The Calculus of Service Availability* e Workbook (SLO/alerting).
  `https://sre.google/static/pdf/calculus_of.pdf`